IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of
Ogasawara et al.
Serial No.: 10/787,005
Filed: February 25, 2004
For: COMPILER DEVICE, PROGRAM, AND RECORDING MEDIUM

Date: December 7, 2004
Group Art Unit: 2655
Examiner: Not yet assigned
Docket No.: JP920030021US1

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMISSION OF PRIORITY DOCUMENT

Sir:

Enclosed herewith is a certified copy of Japanese Application No. 2003-49414 filed February 26, 2003, in support of applicant's claim to priority under 35 U.S.C. 119.

Respectfully submitted,

By _____
Louis P. Herzberg
Reg. No. 41,500
Phone No. (914) 945-2885

IBM Corporation
Intellectual Property Law Dept.
P.O. Box 218
Yorktown Heights, NY 10598

日　本　国　特　許　庁
JAPAN　PATENT　OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application:　２００３年　２月２６日

出願番号
Application Number:　特願２００３－０４９４１４

ST.10/C]:　［ＪＰ２００３－０４９４１４］

出願人
pplicant(s):　インターナショナル・ビジネス・マシーンズ・コーポレーション

２００３年　５月２０日

特許庁長官
Commissioner,
Japan Patent Office　太田信一郎

出証番号　出証特２００３－３０３７０２８

【書類名】　特許願

【整理番号】　JP9030021

【提出日】　平成15年 2月26日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F　9/45

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　竹内　幹雄

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　小野寺　民也

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　小笠原　武史

【特許出願人】

【識別番号】　390009531

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【弁理士】

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【弁理士】

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100108501

【弁理士】

【氏名又は名称】　上野　剛史

【復代理人】

【識別番号】　100104156

【弁理士】

【氏名又は名称】　龍華　明裕

【手数料の表示】

【予納台帳番号】　053394

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0207860

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　コンパイラ装置、コンパイラプログラム、及び記録媒体

【特許請求の範囲】

【請求項１】　文字列を操作するプログラムを最適化するコンパイラ装置であって、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と

を備えるコンパイラ装置。

【請求項２】　前記複数の追加命令によって文字列を追加した後に初めて前記文字列変数を参照する参照命令を検出する参照命令検出部を更に備え、

前記追加コード生成部は、前記追加コードを、全ての前記格納コードより後、かつ前記参照命令より先に実行される個所に生成する請求項１記載のコンパイラ装置。

【請求項３】　前記追加命令検出部は、前記追加命令として、文字列を追加する処理が許可されていない不変文字列変数を、文字列を追加する処理が許可された可変文字列変数に変換する命令と、前記可変文字列変数に前記追加文字列を追加する命令と、前記可変文字列変数を不変文字列変数に変換する命令との組合せを検出する請求項１記載のコンパイラ装置。

【請求項４】　文字列を操作するプログラムを最適化するコンパイラ装置であって、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する

複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と

を備えるコンパイラ装置。

【請求項５】　文字列を操作するプログラムを最適化するコンパイラ装置であって、

文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変化命令を検出する不変化命令検出部と、

前記不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部と、

前記不変化命令から前記可変化命令までの間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部と

を備えるコンパイラ装置。

【請求項６】　前記不変文字列変数に格納された文字列が参照されない場合に、前記命令除去部は、前記不変化命令を更に除去する請求項５記載のコンパイラ装置。

【請求項７】　前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている

文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う請求項６記載のコンパイラ装置。

【請求項８】　前記可変化命令検出部は、可変文字列変数として用いられる記憶領域を確保する命令と、当該可変文字列変数に、前記不変文字列変数に格納された文字列を追加する命令との組合せを、前記可変化命令として検出する請求項５記載のコンパイラ装置。

【請求項９】　前記可変化命令検出部により検出された前記可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う部分冗長除去部を更に備え、

前記部分冗長性の前記除去処理が実行されたプログラムにおいて、前記不変化命令及び前記可変化命令の間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記命令除去部は、前記可変化命令を除去する請求項５記載のコンパイラ装置。

【請求項１０】　前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う請求項９記載のコンパイラ装置。

【請求項１１】　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と
して機能させるコンパイラプログラム。

【請求項１２】　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、

前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、

前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部と
して機能させるコンパイラプログラム。

【請求項１３】　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、

前記コンピュータを、

文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変化命令を検出する不変化命令検出部と、

前記不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部と、

前記不変化命令から前記可変化命令までの間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令によ

り変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部と
して機能させるコンパイラプログラム。

【請求項１４】　請求項１１から１３の何れかに記載のコンパイラプログラムを記録した記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、コンパイラ装置、コンパイラプログラム、及び記録媒体に関する。特に本発明は、文字列を操作するプログラムを最適化するコンパイラ装置、コンパイラプログラム、及び記録媒体に関する。

【０００２】

【従来の技術】

近年、プログラムの保守性や堅牢性を高めるために、プログラマにとって直感的な演算子を記載することのできるプログラム言語が普及している。例えば、Ｊａｖａ（登録商標）言語において、プログラマは、文字列変数に文字列を追加する処理を、「＋」演算子により、簡易に記載することができる。

また、従来、コンパイラにおいて、部分冗長性を除去する技術（非特許文献１参照。）及び部分不用代入文を除去する技術（非特許文献２参照。）が用いられている。

【０００３】

【非特許文献１】

Ｊ．Ｋｎｏｏｐ、Ｏ．Ｒｕｔｈｉｎｇ　ａｎｄ　Ｂ．Ｓｔｅｆｆｅｎ、Ｌａｚｙ　ｃｏｄｅ　ｍｏｔｉｏｎ、Ｉｎ　ＰＬＤＩ‘９２、ｐ．２２４－２３４、１９９２．邦題「Ｌａｚｙコード移動」

【０００４】

【非特許文献２】

J. Knoop、O. Ruthing and B. Steffen、Partial dead code elimination、In PLDI '94、p. １４７－１５８、１９９４．邦題「部分不用代入文除去」

【０００５】

【発明が解決しようとする課題】

文字列の追加処理は、実際には、メモリ領域の確保及びデータの複写等の、多数の処理の集まりにコンパイルされる。しかしながら、同一の文字列変数に、複数の文字列が順次追加される場合には、メモリ領域の確保及びデータの複写等が冗長となる場合がある。更に、これらの冗長な命令は、部分冗長性の除去及び部分不用代入文の除去の技術をそのまま適用しても、適切に除去されない場合がある。例えば、サーバ用プログラムにおいて、文字列の追加処理が頻繁に用いられる場合には、サーバの動作効率を低めてしまっていた。

そこで本発明は、上記の課題を解決することのできるコンパイラ装置、コンパイラプログラム、及び記録媒体を提供することを目的とする。この目的は特許請求の範囲における独立項に記載の特徴の組み合わせにより達成される。また従属項は本発明の更なる有利な具体例を規定する。

【０００６】

【課題を解決するための手段】

即ち、本発明の第１の形態によると、文字列を操作するプログラムを最適化するコンパイラ装置であって、プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、プログラムにおける、文字列変数を参照する命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置、コンパイラプログラム、及びコンパイラプログラムを記録した記録媒体を提供する。

なお上記の発明の概要は、本発明の必要な特徴の全てを列挙したものではなく

、これらの特徴群のサブコンビネーションも又発明となりうる。

【０００７】

【発明の実施の形態】

以下、発明の実施の形態を通じて本発明を説明するが、以下の実施形態は特許請求の範囲にかかる発明を限定するものではなく、又実施形態の中で説明されている特徴の組み合わせの全てが発明の解決手段に必須であるとは限らない。

【０００８】

（第１実施形態）

図１は、コンパイラ装置１０のブロック図を示す。コンパイラ装置１０は、例えば、Ｊａｖａ（登録商標）のＪｕｓｔ－Ｉｎ－Ｔｉｍｅコンパイラであり、文字列を格納する文字列変数に文字列を追加する追加命令を最適化することを目的とする。コンパイラ装置１０は、追加命令を検出する追加命令検出部１００と、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部１１０と、文字列変数を参照する参照命令を検出する参照命令検出部１２０と、バッファに格納された追加文字列を文字列変数に追加する追加コード生成部１３０とを備える。

【０００９】

追加命令検出部１００は、コンパイル対象のプログラムを取得すると、当該プログラムにおいて、同一の文字列変数に文字列を追加する複数の追加命令を検出し、検出結果を格納コード生成部１１０及び参照命令検出部１２０に送る。格納コード生成部１１０は、追加命令の検出結果に基づき、コンパイル対象のプログラムにおける、追加命令検出部１００によって検出された複数の追加命令のそれぞれに代えて、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する。そして、格納コード生成部１１０は、格納コードが生成された当該プログラムを追加コード生成部１３０に送る。

【００１０】

参照命令検出部１２０は、追加命令の検出結果に基づき、コンパイル対象のプログラムにおいて、複数の追加命令によって文字列を追加した後初めて文字列変数を参照する参照命令を検出し、検出結果を追加コード生成部１３０に送る。追

加コード生成部１３０は、参照命令の検出結果に基づき、格納コード生成部１１０から受け取ったプログラムにおける、全ての格納コードより後、かつ参照命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。そして、追加コード生成部１３０は、追加コードが生成された当該プログラムを、コンパイル結果のプログラムとして出力する。なお、コンパイラ装置１０は、本図の各部材が行う処理より先又は後に、他の最適化を行ってもよい。

【００１１】

図２は、コンパイラ装置１０の動作フローを示す。コンパイラ装置１０は、例えば、Ｊａｖａ（登録商標）における、コンパイル対象のプログラムを含むクラスファイルを取得する（Ｓ２００）。そして、追加命令検出部１００は、同一の文字列変数に文字列を追加する複数の追加命令を検出する（Ｓ２１０）。格納コード生成部１１０は、コンパイル対象のプログラムにおいて、追加命令検出部１００において検出された複数の追加命令のそれぞれに代えて、追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する（Ｓ２２０）。参照命令検出部１２０は、複数の追加命令によって文字列を追加した後に初めて文字列変数を参照する参照命令を検出する（Ｓ２３０）。そして、追加コード生成部１３０は、全ての格納コードより後、かつ参照命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する（Ｓ２４０）。

【００１２】

図３（ａ）は、最適化対象のプログラムのソースコードの一例を示す。１行目の命令は、文字列変数の一例であるｓに、空の文字列“”を代入する命令である。２から４行目の命令のそれぞれは、変数ｃに格納された追加文字列をｓと連結（ｃｏｎｃａｔｅｎａｔｅ）し、その結果をｓに代入する命令、即ち、変数ｃに格納された追加文字列をｓに追加する追加命令である。なお、２から４行目の命令のそれぞれは、ループにより繰り返し実行される、ソースコード上で同一の追加命令であってもよい。５行目の命令は、２から４行目のそれぞれの追加命令によって文字列を追加した後に初めてｓを参照する参照命令である。

【００１３】

図３（ｂ）は、プログラムがコンパイルされた結果を示す。より詳細には、例えば、本図は、Ｊａｖａ（登録商標）におけるバイトコードコンパイラが、図３（ａ）に示したソースコードを、バイトコードにコンパイルした結果の一例を示す。なお、実際には、バイトコードコンパイラは、各命令を数値データで示したバイトコードを出力するが、本図は、説明の都合上バイトコードの意味を示す擬似コードを示す。なお、以降に説明する図５及び図８から１２においても、説明の都合上、命令の意味を示す擬似コードを用いる。

【００１４】

１行目の命令は、文字列の追加が許可されていない不変文字列変数ｓを、文字列の追加が許可された可変文字列変数ｓｂに変換する命令であり、例えば、文字列の追加が許可された可変文字列変数ｓｂを格納する領域をメモリに確保し、文字列の追加が許可されていない不変文字列変数ｓに格納される文字列を、ｓｂに複写する。２行目の命令は、ｓｂに、変数ｃに格納された追加文字列を追加する命令である。なお、変数ｃがキャラクタ変数でない場合には、２行目の命令に加え、更に、ｖａｌｕｅＯｆメソッドが実行され、その過程で不変文字列変数が更に確保される場合がある。３行目の命令は、ｓｂを、不変文字列ｓに変換する命令である。以降同様に、４行目及び７行目の命令のそれぞれは、１行目の命令と略同一であり、５行目及び８行目の命令のそれぞれは、２行目の命令と略同一であり、６行目及び９行目の命令のそれぞれは、３行目の命令と略同一である。また、１０行目の命令は、不変文字列変数ｓを参照する命令である。

【００１５】

このように、不変文字列変数ｓに、新たに文字列ｃを追加する命令は、メモリ上の領域確保及びデータの複写を含む複数の命令にコンパイルされる。本図のプログラムによると、最終的に用いられない変数の領域確保が、ループで繰り返されてしまう。例えば、１行目、４行目、及び７行目において、最終的に不要な変数ｓｂの領域確保が繰り返される。更に、新たに確保されたメモリ領域を初期化する処理が必要となり、効率が悪い。また、コンパイラ装置１０が動的メモリ管理を行う場合には、不要なメモリ領域は、ガーベージコレクション（Ｇａｒｂａ

ｇｅ　Ｃｏｌｌｅｃｔｉｏｎ）により頻繁に開放され、効率が悪い。

【００１６】

ここで、文字列を追加する処理が許可された可変文字列変数（ｍｕｔａｂｌｅ　ｓｔｒｉｎｇ　ｖａｒｉａｂｌｅ）とは、プログラム言語の仕様により、文字列の追加操作が許可されている変数であり、一例としては、Ｊａｖａ（登録商標）言語におけるＳｔｒｉｎｇＢｕｆｆｅｒオブジェクトのインスタンスである。そして、プログラム言語がオブジェクト指向言語の場合、可変文字列変数を含む可変文字列オブジェクトは、可変文字列変数に格納される文字列に新たに文字列を追加する追加メソッド、例えば、ａｐｐｅｎｄメソッドを有していてもよい。

【００１７】

一方、文字列を追加する処理が許可されていない不変文字列変数（ｉｍｍｕｔａｂｌｅ　ｓｔｒｉｎｇ　ｖａｒｉａｂｌｅ）とは、プログラム言語の仕様により、文字列の追加操作が許可されていない変数であり、一例としては、Ｊａｖａ（登録商標）言語におけるＳｔｒｉｎｇオブジェクトのインスタンスである。そして、コンパイラ装置１０は、不変文字列変数に格納された文字列が変化しないと仮定してコンパイル処理を行えるので、効率的となる。例えば、コンパイラ装置１０は、不変文字列変数へのアクセスに、排他制御を必要としないので、高速に実行させることができる。

【００１８】

図３（ｃ）は、コンパイラ装置１０による最適化を終えたプログラムの一例を示す。追加命令検出部１００は、追加命令として、不変文字列変数ｓを可変文字列変数ｓｂに変換する図３（ｂ）の１行目の命令と、可変文字列変数ｓｂに追加文字列ｃを追加する図３（ｂ）の２行目の命令と、可変文字列変数ｓｂを不変文字列変数ｓに変換する図３（ｂ）の３行目の命令との組合せを検出する。そして、格納コード生成部１１０は、追加命令に代えて、追加文字列ｃのデータをバッファに格納する格納コード、例えば、図３（ｃ）の２行目の命令を生成する。

【００１９】

ここで、バッファとは、プログラムを実行するスレッド毎に、予め確保されたメモリ上の領域である。バッファは、本図のプログラムの実行に加え、同一のス

レッドによる他の処理に用いられるので、文字のデータサイズより充分に大きいことが望ましい。一例としては、バッファのサイズは、１Ｍｂｙｔｅであってもよいし、１０Ｍｂｙｔｅであってもよい。また、本図で示した変数ｂｕｆは、バッファにおける空き領域の先頭のアドレスを格納する変数である。

【００２０】

追加コード生成部１３０は、不変文字列変数ｓを参照する７行目の命令より先に実行される個所、例えば、５行目に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。例えば、追加コード生成部１３０は、追加コードとして、追加文字列を格納しているバッファのアドレスを、不変文字列変数を含むオブジェクトのインスタンス変数（ｓ．ｖａｌｕｅ）に代入する命令を生成する。この結果、７行目の命令は、追加処理を終えた後の文字列変数を適切に参照できる。更に、本図に示したプログラムを実行するスレッドが、本図に示したプログラム以降に更に他のプログラムを実行する場合に備え、追加コード生成部１３０は、バッファの空き領域を示す情報を更新する処理を行うコードを生成することが望ましい。例えば、追加コード生成部１３０は、変数ｂｕｆの値を更新する命令を６行目に生成することにより、後に実行されるコードに、バッファの空き領域を適切に知らせることができる。

【００２１】

このように、コンパイラ装置１０は、可変文字列変数の領域をメモリに確保する命令及び文字列のデータを複写する命令を除去することにより、文字列を追加する追加命令を最適化することができる。

なお、本実施形態において、コンパイラ装置１０は、例えば、バイトコードを実行コードにコンパイルするＪｕｓｔ－Ｉｎ－Ｔｉｍｅコンパイラである。これに代えて、コンパイラ装置１０は、ソースプログラムをバイトコードにコンパイルするバイトコードコンパイラであってもよい。この場合、追加命令検出部１００は、ソースプログラムにおいて、文字列変数に文字列を追加する「＋」演算子を、追加命令として検出する。そして、格納コード生成部１１０は、「＋」演算子に代えて、格納コードを生成する。即ち、コンパイラ装置１０は、図３（ａ）に示したソースプログラムを入力とし、図３（ｂ）のプログラムを介さずに、図

３（ｃ）と略同一の処理を行うバイトコードを出力してもよい。

【００２２】

図４は、変形例において、コンパイラ装置１０がプログラムを最適化する一例を示す。本例におけるコンパイラ装置１０は、図３（ｃ）に示したプログラムに代えて、本図に示したプログラムを生成してもよい。

【００２３】

コンパイラ装置１０は、バッファを初期化する１行目の命令を生成する。バッファｂｕｆは、連結リスト構造であり、連結リストの各要素は、追加文字列を１づつ格納することができる。格納コード生成部１１０は、図３（ａ）における追加命令に代えて、追加文字列ｃのデータをバッファに格納する図４の２行目の命令と、連結リストにおける次の要素を示すアドレスを取得する３行目の命令とを、格納コードとして生成する。以降同様に、格納コード生成部１１０は、複数の追加命令のそれぞれに代えて、４行目及び６行目において、格納コードを生成する。そして、追加コード生成部１３０は、不変文字列変数ｓを参照する９行目の命令より先に実行される個所、例えば、８行目に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する。

【００２４】

このように、格納コード生成部１１０は、複数の追加命令のそれぞれに代えて、追加文字列及び追加文字列が格納されているアドレスとの組合せを、連結リスト状に、バッファに格納してもよい。この場合、連結リストをヌルポインタで予め終端しておくことで、コンパイラ装置１０は、メモリアクセス違反を検出することにより、連結リストの終端を検出できる。これにより、各格納コードにおける、連結リストの終端チェック命令が省略できるので、処理を高速化することができる。

【００２５】

図５（ａ）は、第１の他の方法においてプログラムが最適化された結果を示す。より詳細には、本例は、熟練したＪａｖａ（登録商標）プログラマが、文字列操作の実行効率を高めるべく、「＋」演算子を用いることなく、文字列の追加処理を記述した例である。１行目の命令は、可変文字列変数ｓｂを格納する領域を

メモリに確保し、不変文字列変数ｓに格納される文字列を、ｓｂに複写する命令である。２行目の命令は、ｓｂに、変数ｃに格納された追加文字列を追加する命令である。以降同様に、３行目及び４行目の命令のそれぞれは、２行目の命令と略同一である。そして、５行目の命令は、ｓｂを、不変文字列ｓに変換する命令である。また、６行目の命令は、不変文字列変数ｓを参照する命令である。

【００２６】

このように、本例によると、追加命令に代えて、可変文字列変数に文字列を追加するコードを用いるので、図３（ｂ）に示したプログラムより性能が高い。

【００２７】

図５（ｂ）は、第２の他の方法においてプログラムが最適化された結果を示す。本例において、不変文字列変数を含むＳｔｒｉｎｇオブジェクトは、Ｊａｖａ（登録商標）の言語仕様に規定されていない、非公開文字列構築子を有している。非公開文字列構築子によると、新たなＳｔｒｉｎｇオブジェクトを生成すると共に、新たな当該Ｓｔｒｉｎｇオブジェクトに、複数のＳｔｒｉｎｇオブジェクトを連結した文字列を複写することができる。例えば、２から４行目の命令のそれぞれは、新たに不変文字列変数を生成し、当該不変文字列変数に、２つの不変文字列変数の内容を複写する命令である。

【００２８】

これにより、可変文字列変数への変換処理を省くことができるので、本例に示したプログラムは、図３（ｂ）に示したプログラムより性能が高い。しかしながら、２から４行目のそれぞれにおいて、不変文字列変数を格納する領域が確保されるにも関わらず、５行目の命令においては、４行目の命令により生成された不変文字列変数のみを参照するので、依然として冗長である。

【００２９】

これに対し、第１実施形態におけるコンパイラ装置１０は、冗長なメモリ確保命令や変換処理を除去することにより、文字列を操作する命令を適切に最適化することができる。

【００３０】

（第２実施形態）

図６は、コンパイラ装置６０のブロック図を示す。コンパイラ装置６０は、例えば、Ｊａｖａ（登録商標）のＪｕｓｔ－Ｉｎ－Ｔｉｍｅコンパイラであり、文字列を格納する文字列変数に文字列を追加する追加命令を最適化することを目的とする。コンパイラ装置６０は、不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部の一例である意味復元部６００と、部分冗長性の除去処理を実行する部分冗長除去部６１０と、可変文字列変数を不変文字列変数に変換する不変化命令を検出する不変化命令検出部６２０と、可変化命令を除去する命令除去部６３０と、検出した可変化命令を実行可能な状態に変換する逆意味復元部６４０と、コンパイル対象のプログラムを実行することによりプロファイルデータを取得する計測コード付きプログラム実行部６４２と、プロファイルデータを格納する不変文字列変数最終長データベース６４５と、可変文字列変数の初期長を指定する可変文字列変数初期長指定部６５０とを備える。

【００３１】

意味復元部６００は、コンパイル対象のプログラムを取得すると、不変文字列変数を可変文字列変数に変換する可変化命令を検出する。より詳細には、例えば、意味復元部６００は、可変化命令の機能を実現する複数の命令の組合せを検出し、当該命令の組合せを、可変化命令を示す擬似命令に変換する。そして、意味復元部６００は、変換した擬似命令を含むプログラムを、部分冗長除去部６１０に送る。

【００３２】

部分冗長除去部６１０は、擬似命令を含むプログラムを意味復元部６００から受け取ると、意味復元部６００により検出された可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う。なお、移動とは、受け取ったプログラムにおける可変化命令を削除し、当該可変化命令を制御フローの合流元に挿入することにより、命令の実行順序を変更することである。そして、部分冗長除去部６１０は、部分冗長性の除去処理を行ったプログラムを不変化命令検出部６２０に送る。

【００３３】

不変化命令検出部６２０は、部分冗長除去部６１０から受け取ったプログラム

において、可変文字列変数を不変文字列変数に変換する不変化命令を検出する。そして、不変化命令検出部６２０は、部分冗長除去部６１０から受け取ったプログラムを、検出結果と共に、命令除去部６３０に送る。

【００３４】

命令除去部６３０は、代数簡約部６３４と、部分不用代入文除去部６３８を有する。代数簡約部６３４は、不変化命令検出部６２０からプログラムを受け取ると、代数簡約的手法により、可変化命令を除去する。例えば、代数簡約部６３４は、「可変文字列変数ｔを不変文字列変数に変換した結果がｓであるならば、不変文字列変数ｓを変換した結果はｔになる」という代数的性質を用いて、可変化命令を除去する。具体的には、代数簡約部６３４は、不変化命令から可変化命令までの間に実行される命令が、不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ可変化命令から可変化命令により変換された可変文字列変数の使用の間に実行される命令が、不変化命令の変換元の可変文字列変数と、可変化命令により変換された可変文字列変数の何れも書き換えない場合に、可変化命令を除去する。そして、代数簡約部６３４は、可変化命令より後で、不変化命令の変換元となる可変文字列変数を、可変化命令により変換された可変文字列変数として使用させる。その後、代数簡約部６３４は、可変化命令を除去したプログラムを部分不用代入文除去部６３８に送る。

【００３５】

部分不用代入文除去部６３８は、部分不用代入文除去の処理として、以下の処理を行う。部分不用代入文除去部６３８は、不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動する。そして、部分不用代入文除去部６３８は、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する。そして、部分不用代入文除去部６３８は、部分不用代入文除去の処理を行ったプログラムを、逆意味復元部６４０に送る。

【００３６】

逆意味復元部６４０は、不変化命令検出部６２０により変換された不変化命令を、不変化命令の機能を実現する複数の命令の組合せに戻す変換処理を行う。そ

して、逆意味復元部６４０は、変換結果のプログラムを可変文字列変数初期長指定部６５０に送る。

【００３７】

計測コード付きプログラム実行部６４２は、コンパイラ装置１０が行うコンパイルの効率を高めるべく、コンパイル対象のプログラムを予め実行し、実行の状態を計測する。例えば、計測コード付きプログラム実行部６４２は、文字列変数に実際に格納された文字列の長さを計測し、計測結果を不変文字列変数最終長データベース６４５に格納する。

【００３８】

可変文字列変数初期長指定部６５０は、コンパイル対象のプログラムを逆意味復元部６４０から受け取り、可変文字列変数に実際に格納された文字列の長さを示すサイズ情報を不変文字列変数最終長データベース６４５から受け取る。そして、可変文字列変数初期長指定部６５０は、可変文字列変数が新たに生成される場合には、サイズ情報が示す長さの領域を予め確保するように、可変文字列変数の初期長を設定する。これにより、コンパイラ装置１０は、プログラムの実行時に、可変文字列変数のメモリ領域を伸張させることを防ぐことができる。そして、可変文字列変数初期長指定部６５０は、可変文字列変数の初期長を設定したプログラムを、コンパイル結果のプログラムとして出力する。

なお、コンパイラ装置６０は、計測コード付きプログラム実行部６４２と、不変文字列変数最終長データベース６４５と、可変文字列変数初期長指定部６５０とを備えなくともよい。この場合、コンパイラ装置６０は、コンパイラ装置６０の設計者により予め設定された値を、可変文字列変数の初期長としてもよい。

【００３９】

図７は、コンパイラ装置６０の動作フローを示す。意味復元部６００は、可変化命令の機能を実現する複数の命令の組合せを検出し、当該命令の組合せを、可変化命令を示す擬似命令に変換する意味復元を行うことにより、不変文字列変数を可変文字列変数に変換する可変化命令を検出する（Ｓ７００）。部分冗長除去部６１０は、擬似命令を含むプログラムを意味復元部６００から受け取ると、意味復元部６００により検出された可変化命令を、当該可変化命令より前において

合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う（Ｓ７１０）。不変化命令検出部６２０は、部分冗長除去部６１０から受け取ったプログラムにおいて、可変文字列変数を不変文字列変数に変換する不変化命令を検出する（Ｓ７２０）。

【００４０】

代数簡約部６３４は、代数簡約的手法により、可変化命令を除去する。例えば、代数簡約部６３４は、不変化命令から可変化命令までの間に実行される命令が、不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ可変化命令から可変化命令により変換された可変文字列変数の使用の間に実行される命令が、不変化命令の変換元の可変文字列変数と、可変化命令により変換された可変文字列変数の何れも書き換えない場合に、可変化命令を除去する。そして、代数簡約部６３４は、不変化命令の変換元となる可変文字列変数を、除去した可変化命令が存在していた個所より後の部分で、可変化命令により変換された可変文字列変数として使用させる（Ｓ７３０）。

【００４１】

部分不用代入文除去部６３８は、不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれにおいて実行させるように、命令の実行順序を変更する。そして、部分不用代入文除去部６３８は、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去の処理を行う（Ｓ７４０）。

【００４２】

逆意味復元部６４０は、意味復元部６００により変換された可変化命令を、可変化命令の機能を実現する複数の命令の組合せに戻す変換処理を行う（Ｓ７５０）。可変文字列変数初期長指定部６５０は、計測コード付プログラム実行部６４２により予め計測され、可変文字列変数に実際に格納された文字列の長さを示すサイズ情報に基づき、可変文字列変数の初期長を設定する（Ｓ７６０）。

【００４３】

図８は、コンパイラ装置６０が最適化する対象のプログラムの一例を示す。本

図のプログラムは、１から３行目の命令を含む基本ブロック８００と、４から９行目の命令を含む基本ブロック８１０と、１０行目の命令を含む基本ブロック８２０とを有する。

【００４４】

１行目の命令は、不変文字列変数の一例であるｓに、空の文字列“”を代入する命令である。２行目の命令は、整数型変数のｉに、ｉの初期値を示す変数ｍｉｎを代入する命令である。３行目の命令は、ｉと、ｉの最終値を示すｍａｘとを比較し、ｉがｍａｘ以上の場合に、基本ブロック８２０を実行し、ｉがｍａｘより小さい場合に、基本ブロック８１０を実行する分岐命令である。

【００４５】

４行目の命令は、可変化文字列変数を含むＳｔｒｉｎｇＢｕｆｆｅｒオブジェクトｔのメモリ上の領域を、新たに確保する命令である。５行目の命令は、ｓに格納されている文字列をｔに追加する命令である。６行目の命令は、文字列型配列変数Ａにおける、ｉ番目の文字列を、ｔに追加する命令である。７行目の命令は、ｔを、不変文字列変数のｓに変換する不変化命令である。８行目の命令は、ｉをインクリメントする命令である。９行目の命令は、ｉと、ｉの最終値を示すｍａｘとを比較し、ｉがｍａｘ以上の場合に、基本ブロック８２０を実行し、ｉがｍａｘより小さい場合に、再度基本ブロック８１０を実行する分岐命令である。

【００４６】

１０行目の命令は、ｓに格納されている文字列を、標準出力、例えば、ディスプレイ画面のコンソールウィンドウに出力する命令である。

【００４７】

図９は、意味復元部６００が可変化命令を検出したプログラムの一例を示す。意味復元部６００は、図８における４行目及び５行目の命令、即ち、可変文字列変数として用いられる記憶領域を確保する命令と、可変文字列変数に不変文字列変数を複写する命令との組合せを、可変化命令として検出する。そして、意味復元部６００は、図９の４行目に示したように、検出した可変化命令を、ｓからｔへの可変化命令である旨を示す擬似命令に置き換える。即ち、意味復元部６００

は、可変化命令の処理を構成する複数の命令の組合せを検出し、当該複数の命令が示す処理の意味を復元する。

【００４８】

なお、意味復元部６００は、プログラムにおいて、連続して配置された４行目及び５行目の命令を検出することにより、可変化命令を検出する。これに代えて、意味復元部６００は、４行目及び５行目の命令の間に他の命令が配置されている場合であっても、新たに確保された可変文字列変数に対する最初の操作が不変文字列の追加であれば、４行目及び５行目の命令を、不変化命令として検出してもよい。

【００４９】

図１０は、部分冗長除去部６１０が部分冗長性を除去したプログラムの一例を示す。部分冗長除去部６１０は、意味復元部６００により検出された可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動するべく、４行目の可変化命令を、基本ブロック８１０から除去すると共に、基本ブロック８３０及び基本ブロック８４０に挿入する。なお、より詳細には、部分冗長除去部６１０は、部分冗長性の除去処理として、非特許文献２に示した技術等と略同一の処理を行ってもよい。例えば、既存の技術によれば、部分冗長除去部６１０は、可変化命令を、当該可変化命令による変換元である変数ｓに値を代入する命令より先に実行させないという条件の元、可変化命令をより先に移動する。

【００５０】

図１１は、命令除去部６３０が可変化命令を除去したプログラムの一例を示す。代数簡約部６３４は、代数簡約的手法により、基本ブロック８４０に移動した可変化命令を削除する。即ち、基本ブロック８１０から基本ブロック８４０に至る制御フローにおいて、図１０における６行目の不変化命令及び１０行目の可変化命令の間に実行される命令が、可変文字列変数ｔに格納された文字列を書き換えず、かつ可変化命令から可変化命令により変換された可変文字列変数の使用の間に実行される命令が、不変化命令の変換元の可変文字列変数と、可変化命令により変換された可変文字列変数の何れも書き換えないので、代数簡約部６３４は

、基本ブロック８４０における可変化命令を除去する。そして、代数簡約部６３４は、可変化命令より後で、不変化命令の変換元となる可変文字列変数を、可変化命令により変換された可変文字列変数として使用させる（１０行目の命令）。

【００５１】

より詳細には、代数簡約部６３４は、変数に対する値の定義と、変数に格納された値の使用との関係の解析結果、例えば、ＵＤ、ＤＵチェインを用いて、可変文字列変数ｔに格納された文字列が書き換わるか否かを判断する。更に、代数簡約部６３４は、可変文字列変数ｔが、他のスレッドからアクセスされないことを更に条件として、可変文字列変数ｔに格納された文字列が書き換わらないと判断する。一例としては、代数簡約部６３４は、プログラムが単一のスレッドにより実行されることを条件としてもよいし、可変文字列変数ｔへのポインタを他のスレッドに通知していないことをエスケープアナリシスにより解析してもよい。

【００５２】

図１２は、命令除去部６３０が不変化命令を除去したプログラムの一例を示す。部分不用代入文除去部６３８は、部分不用代入文の除去処理として、以下の処理を行う。部分不用代入文除去部６３８は、図１１の６行目の不変化命令を、不変化命令より後に実行される分岐命令、例えば７行目の命令の分岐先のそれぞれにおいて実行させる。即ち、部分不用代入文除去部６３８は、図１１の６行目の不変化命令を、基本ブロック８１０から除去すると共に、基本ブロック８４０及び基本ブロック８５０に移動する。そして、部分不用代入文除去部６３８は、基本ブロック８４０に移動した不変化命令の変換先である不変文字列変数ｓに格納されている文字列が、基本ブロック８４０以降に実行される部分において参照されないので、基本ブロック８４０に移動した当該不変化命令を除去する。

【００５３】

なお、より詳細には、部分不用代入文除去部６３８は、部分不用代入文の除去処理として、非特許文献１に示した技術等と略同一の処理を行ってもよい。例えば、部分不用代入文除去部６３８は、不変化命令を、当該不変化命令による変換先である変数ｓを参照する命令より後に実行させないという条件の元、不変化命令をより後に移動する。

【００５４】

図１３は、コンパイラ装置１０又はコンパイラ装置６０のハードウェア構成の一例を示す。第１の実施形態又は変形例に係るコンパイラ装置１０並びに第２の実施形態に係るコンパイラ装置６０は、ホストコントローラ１０８２により相互に接続されるＣＰＵ１０００、ＲＡＭ１０２０、グラフィックコントローラ１０７５、及び表示装置１０８０を有するＣＰＵ周辺部と、入出力コントローラ１０８４によりホストコントローラ１０８２に接続される通信インターフェイス１０３０、ハードディスクドライブ１０４０、及びＣＤ－ＲＯＭドライブ１０６０を有する入出力部と、入出力コントローラ１０８４に接続されるＲＯＭ１０１０、フレキシブルディスクドライブ１０５０、及び入出力チップ１０７０を有するレガシー入出力部とを備える。

【００５５】

ホストコントローラ１０８２は、ＲＡＭ１０２０と、高い転送レートでＲＡＭ１０２０をアクセスするＣＰＵ１０００及びグラフィックコントローラ１０７５とを接続する。ＣＰＵ１０００は、ＲＯＭ１０１０及びＲＡＭ１０２０に格納されたコンパイラプログラムに基づいて動作し、各部の制御を行う。グラフィックコントローラ１０７５は、ＣＰＵ１０００等がＲＡＭ１０２０内に設けたフレームバッファ上に生成する画像データを取得し、表示装置１０８０上に表示させる。これに代えて、グラフィックコントローラ１０７５は、ＣＰＵ１０００等が生成する画像データを格納するフレームバッファを、内部に含んでもよい。

【００５６】

入出力コントローラ１０８４は、ホストコントローラ１０８２と、比較的高速な入出力装置である通信インターフェイス１０３０、ハードディスクドライブ１０４０、及びＣＤ－ＲＯＭドライブ１０６０を接続する。通信インターフェイス１０３０は、ネットワークを介して他の装置と通信する。ハードディスクドライブ１０４０は、コンパイラ装置１０又はコンパイラ装置６０が使用するコンパイラプログラム及びデータを格納する。ＣＤ－ＲＯＭドライブ１０６０は、ＣＤ－ＲＯＭ１０９５からコンパイラプログラム又はデータを読み取り、ＲＡＭ１０２０を介して入出力チップ１０７０に提供する。

【００５７】

また、入出力コントローラ１０８４には、ＲＯＭ１０１０と、フレキシブルディスクドライブ１０５０や入出力チップ１０７０等の比較的低速な入出力装置とが接続される。ＲＯＭ１０１０は、コンパイラ装置１０又はコンパイラ装置６０の起動時にＣＰＵ１０００が実行するブートプログラムや、コンパイラ装置１０又はコンパイラ装置６０のハードウェアに依存するプログラム等を格納する。フレキシブルディスクドライブ１０５０は、フレキシブルディスク１０９０からコンパイラプログラム又はデータを読み取り、ＲＡＭ１０２０を介して入出力チップ１０７０に提供する。入出力チップ１０７０は、フレキシブルディスク１０９０や、例えばパラレルポート、シリアルポート、キーボードポート、マウスポート等を介して各種の入出力装置を接続する。

【００５８】

コンパイラ装置１０又はコンパイラ装置６０に提供されるコンパイラプログラムは、フレキシブルディスク１０９０、ＣＤ－ＲＯＭ１０９５、又はＩＣカード等の記録媒体に格納されて利用者によって提供される。コンパイラプログラムは、記録媒体から読み出され、入出力チップ１０７０を介してコンパイラ装置１０又はコンパイラ装置６０にインストールされ、コンパイラ装置１０又はコンパイラ装置６０において実行される。

【００５９】

コンパイラ装置１０又はコンパイラ装置６０にインストールされて実行されるコンパイラプログラムは、追加命令検出モジュールと、格納コード生成モジュールと、参照命令検出モジュールと、追加コード生成モジュールと、不変化命令検出モジュールと、部分冗長除去モジュールと、可変化命令検出モジュールと、命令除去モジュールと、代数簡約モジュールと、部分不用代入文除去モジュールと、逆意味復元モジュールと、計測コード付きプログラム実行モジュールと、可変文字列変数初期長指定モジュールとを含む。各モジュールがコンパイラ装置１０又はコンパイラ装置６０に働きかけて行わせる動作は、図１から図１２において説明したコンパイラ装置１０又はコンパイラ装置６０における、対応する部材の動作と同一であるから、説明を省略する。

【００６０】

以上に示したコンパイラプログラム又はモジュールは、外部の記憶媒体に格納されてもよい。記憶媒体としては、フレキシブルディスク１０９０、ＣＤ－ＲＯＭ１０９５の他に、ＤＶＤやＰＤ等の光学記録媒体、ＭＤ等の光磁気記録媒体、テープ媒体、ＩＣカード等の半導体メモリ等を用いることができる。また、専用通信ネットワークやインターネットに接続されたサーバシステムに設けたハードディスク又はＲＡＭ等の記憶装置を記録媒体として使用し、ネットワークを介してコンパイラプログラムをコンパイラ装置１０又はコンパイラ装置６０に提供してもよい。

【００６１】

以上の説明から明らかなように、コンパイラ装置は、文字列を追加する命令を最適化することができる。

【００６２】

以上、本発明を実施形態を用いて説明したが、本発明の技術的範囲は上記実施形態に記載の範囲には限定されない。上記実施形態に、多様な変更または改良を加えることができる。そのような変更または改良を加えた形態も本発明の技術的範囲に含まれ得ることが、特許請求の範囲の記載から明らかである。例えば、コンパイラ装置１０は、コンパイラ装置６０の各部材を更に備えてもよい。この場合、コンパイラ装置１０は、予め定められた条件に基づき、第１実施形態及び第２実施形態に示した機能の何れかを選択して処理してもよい。例えば、コンパイラ装置１０は、計測コード付きプログラム実行部６４２により文字列の長さが計測できた場合は、第２実施形態に示した処理を行い、計測できなかった場合は、第１実施形態に示した処理を行ってもよい。また、コンパイラ装置１０は、計測コード付きプログラム実行部６４２により計測された複数の文字列の長さの分散値が、所定値より大きい場合に、第１実施形態に示した処理を行ってもよい。

【００６３】

以上で示した全ての実施形態及び変形例によると、以下の各項目に示すコンパイラ装置、コンパイラプログラム、及び記録媒体が実現される。

【００６４】

（項目１）　文字列を操作するプログラムを最適化するコンパイラ装置であって、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

（項目２）　前記複数の追加命令によって文字列を追加した後に初めて前記文字列変数を参照する参照命令を検出する参照命令検出部を更に備え、前記追加コード生成部は、前記追加コードを、全ての前記格納コードより後、かつ前記参照命令より先に実行される個所に生成する項目１記載のコンパイラ装置。

【００６５】

（項目３）　前記追加命令検出部は、前記追加命令として、文字列を追加する処理が許可されていない不変文字列変数を、文字列を追加する処理が許可された可変文字列変数に変換する命令と、前記可変文字列変数に前記追加文字列を追加する命令と、前記可変文字列変数を不変文字列変数に変換する命令との組合せを検出する項目１記載のコンパイラ装置。

（項目４）　文字列を操作するプログラムを最適化するコンパイラ装置であって、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部とを備えるコンパイラ装置。

【００６６】

（項目５）　文字列を操作するプログラムを最適化するコンパイラ装置であって、文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変化命令を検出する不変化命令検出部と、前記不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部と、前記不変化命令から前記可変化命令までの間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部とを備えるコンパイラ装置。

（項目６）　前記不変文字列変数に格納された文字列が参照されない場合に、前記命令除去部は、前記不変化命令を更に除去する項目５記載のコンパイラ装置。

（項目７）　前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う項目６記載のコンパイラ装置。

【００６７】

（項目８）　前記可変化命令検出部は、可変文字列変数として用いられる記憶領域を確保する命令と、当該可変文字列変数に、前記不変文字列変数に格納された文字列を追加する命令との組合せを、前記可変化命令として検出する項目５記載のコンパイラ装置。

（項目９）　前記可変化命令検出部により検出された前記可変化命令を、当該可変化命令より前において合流する制御フローの合流元のそれぞれに移動する部分冗長性の除去処理を行う部分冗長除去部を更に備え、前記部分冗長性の前記除去処理が実行されたプログラムにおいて、前記不変化命令及び前記可変化命令の間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文

字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記命令除去部は、前記可変化命令を除去する項目５記載のコンパイラ装置。

【００６８】

（項目１０）　前記命令除去部は、前記不変化命令を、当該不変化命令より後に実行される分岐命令の分岐先のそれぞれに移動し、当該分岐命令の分岐先のそれぞれにおいて、不変化命令の変換先である不変文字列変数に格納されている文字列が参照されない場合に、当該不変化命令を除去する部分不用代入文除去を行う項目９記載のコンパイラ装置。

（項目１１）　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記追加文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部として機能させるコンパイラプログラム。

【００６９】

（項目１２）　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、前記プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、前記追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列が格納されているメモリのアドレスをバッファに格納する格納コードを生成する格納コード生成部と、前記プログラムにおける、前記文字列変数を参照する命令より先に実行される個所に、複数の前記アドレスに格納される複数の追加

文字列のそれぞれを前記文字列変数に追加する追加コードを生成する追加コード生成部として機能させるコンパイラプログラム。

（項目１３）　文字列を操作するプログラムをコンピュータにより最適化するコンパイラプログラムであって、前記コンピュータを、文字列を追加する処理が許可された可変文字列変数を、文字列を追加する処理が許可されていない不変文字列変数に変換する不変化命令を検出する不変化命令検出部と、前記不変文字列変数を可変文字列変数に変換する可変化命令を検出する可変化命令検出部と、前記不変化命令から前記可変化命令までの間に実行される命令が、前記不変化命令の変換元の可変文字列変数に格納された文字列を書き換えず、かつ前記可変化命令から前記可変化命令により変換された可変文字列変数の使用の間に実行される命令が、前記不変化命令の変換元の可変文字列変数と、前記可変化命令により変換された可変文字列変数の何れも書き換えない場合に、前記可変化命令を除去し、前記可変化命令より後で、前記不変化命令の変換元となる可変文字列変数を、前記可変化命令により変換された可変文字列変数として使用させる命令除去部として機能させるコンパイラプログラム。

（項目１４）　項目１１から１３の何れかに記載のコンパイラプログラムを記録した記録媒体。

【００７０】

【発明の効果】

上記説明から明らかなように、本発明によれば文字列を操作するプログラムを最適化することができる。

【図面の簡単な説明】

【図１】

図１は、コンパイラ装置１０のブロック図を示す。

【図２】

図２は、コンパイラ装置１０の動作フローを示す。

【図３】

図３（ａ）は、最適化対象のプログラムのソースコードの一例を示す。

図３（ｂ）は、プログラムがコンパイルされた結果を示す。

図３（ｃ）は、コンパイラ装置１０による最適化を終えたプログラムの一例を示す。

【図４】

図４は、変形例において、コンパイラ装置１０がプログラムを最適化する一例を示す。

【図５】

図５（ａ）は、第１の他の方法においてプログラムが最適化された結果を示す。

図５（ｂ）は、第２の他の方法においてプログラムが最適化された結果を示す。

【図６】

図６は、コンパイラ装置６０のブロック図を示す。

【図７】

図７は、コンパイラ装置６０の動作フローを示す。

【図８】

図８は、コンパイラ装置６０が最適化する対象のプログラムの一例を示す。

【図９】

図９は、意味復元部６００が可変化命令を検出したプログラムの一例を示す。

【図１０】

図１０は、部分冗長除去部６１０が部分冗長性を除去したプログラムの一例を示す。

【図１１】

図１１は、命令除去部６３０が可変化命令を除去したプログラムの一例を示す。

【図１２】

図１２は、命令除去部６３０が不変化命令を除去したプログラムの一例を示す。

【図１３】

図１３は、コンパイラ装置１０又はコンパイラ装置６０のハードウェア構成の

一例を示す。

【符号の説明】

１０　コンパイラ装置

６０　コンパイラ装置

１００　追加命令検出部

１１０　格納コード生成部

１２０　参照命令検出部

１３０　追加コード生成部

６００　可変化命令検出部

６１０　意味復元部

６２０　不変化命令検出部

６３０　命令除去部

６３４　代数簡約部

６３８　部分不用代入文除去部

６４０　逆意味復元部

６４２　計測コード付きプログラム実行部

６４５　不変文字列変数最終長データベース

６５０　可変文字列変数初期長指定部

８００　基本ブロック

８１０　基本ブロック

８２０　基本ブロック

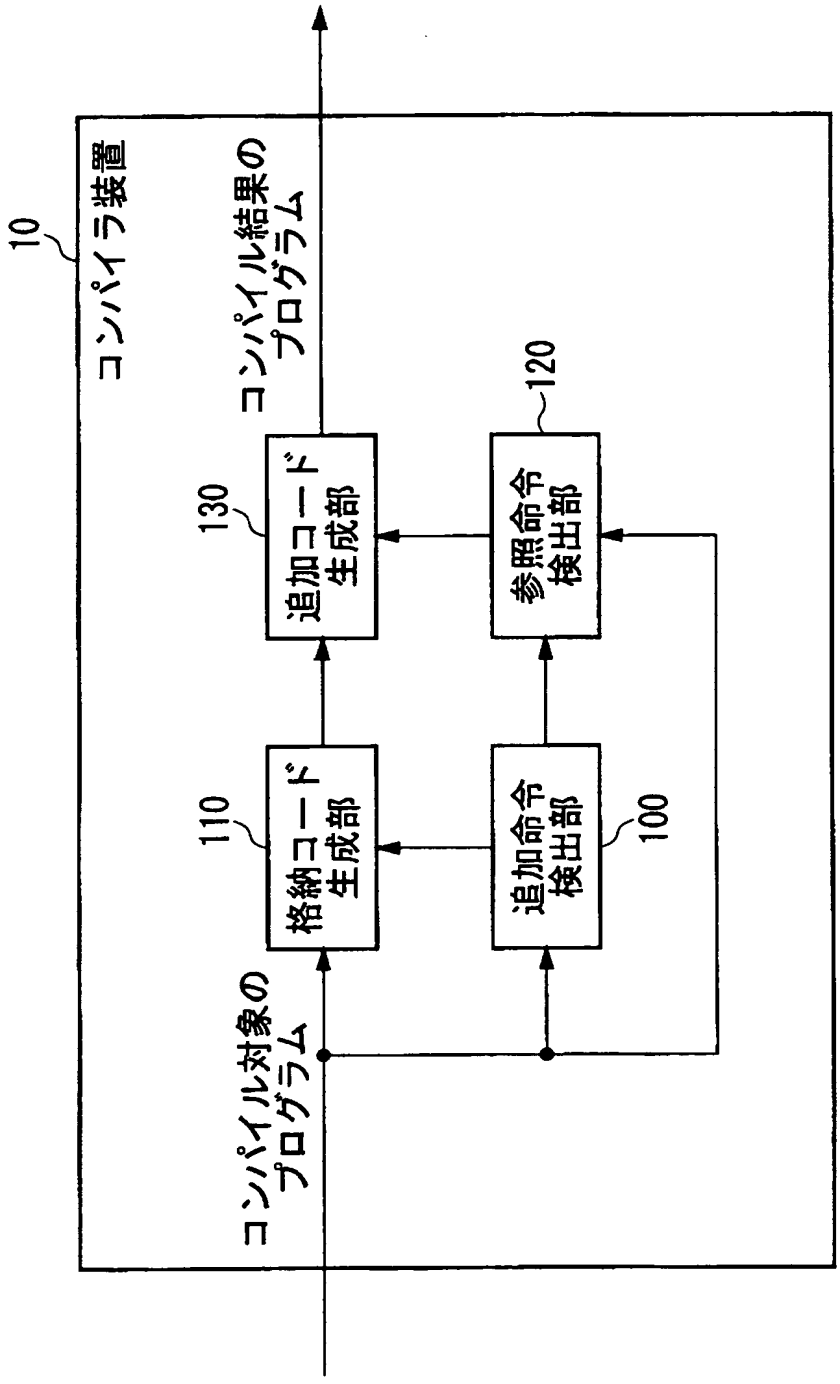
８３０　基本ブロック

８４０　基本ブロック

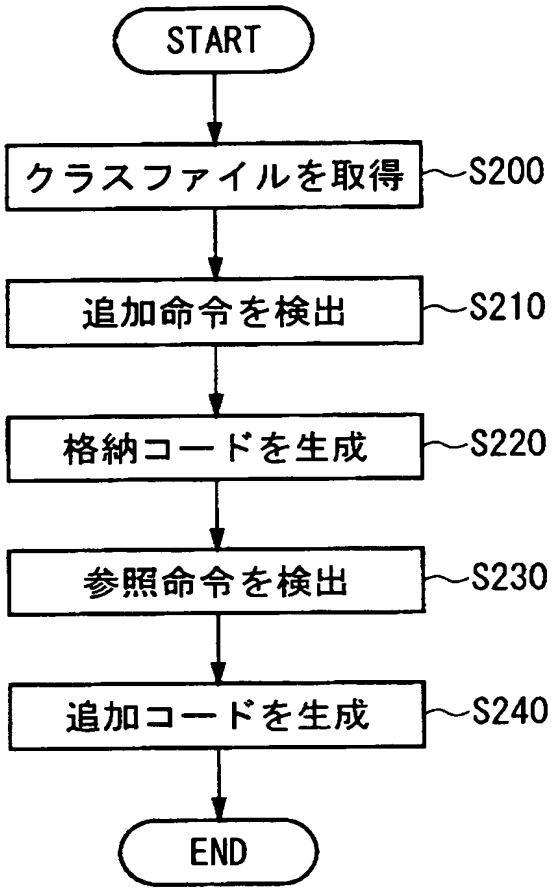
８５０　基本ブロック

【書類名】　図面

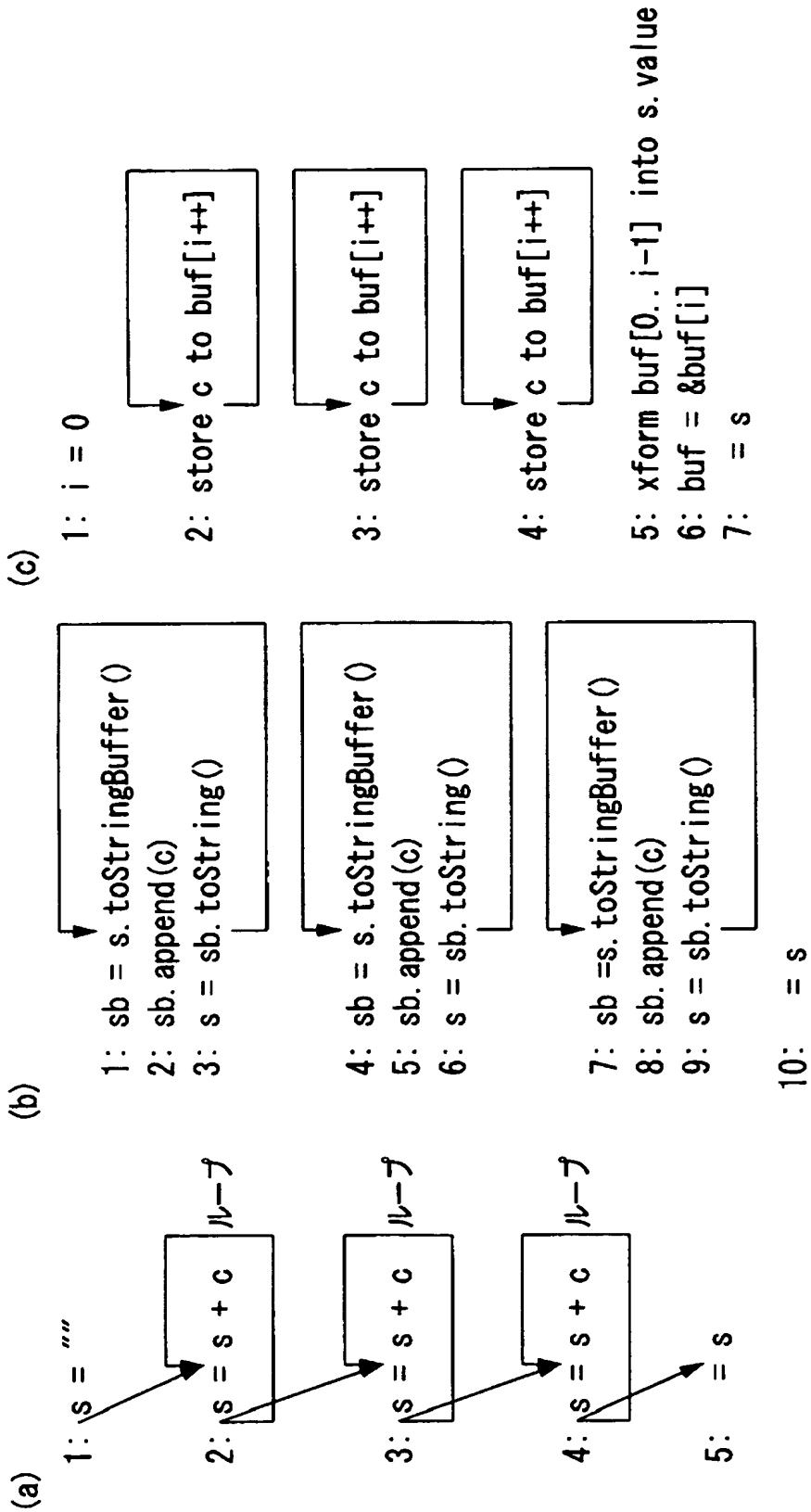
【図１】


10
コンパイラ装置
コンパイル結果のプログラム
120
130
追加コード生成部
参照命令検出部
110
格納コード生成部
追加命令検出部
100
コンパイル対象のプログラム

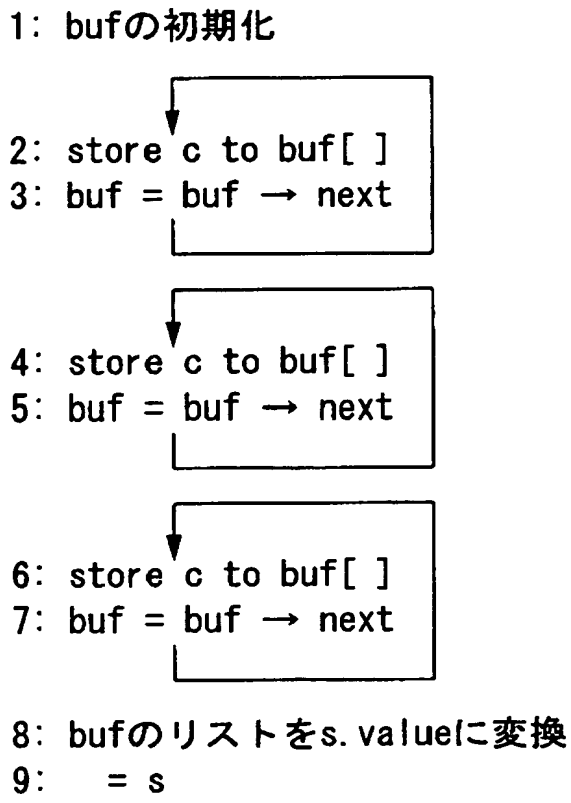
【図２】


START
クラスファイルを取得
S200
追加命令を検出
S210
格納コードを生成
S220
参照命令を検出
S230
追加コードを生成
S240
END

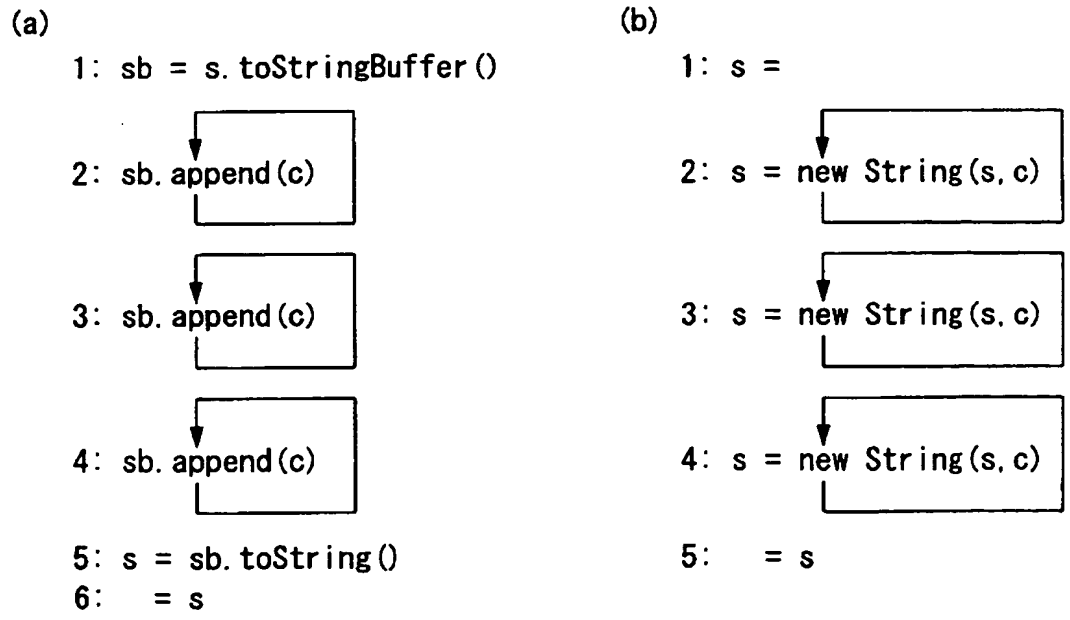
【図３】


(a)
1: s = ""
2: s = s + c
ループ
3: s = s + c
ループ
4: s = s + c
ループ
5: = s
(b)
1: sb = s.toStringBuffer()
2: sb.append(c)
3: s = sb.toString()
4: sb = s.toStringBuffer()
5: sb.append(c)
6: s = sb.toString()
7: sb =s.toStringBuffer()
8: sb.append(c)
9: s = sb.toString()
10: = s
(c)
1: i = 0
2: store c to buf[i++]
3: store c to buf[i++]
4: store c to buf[i++]
5: xform buf[0..i-1] into s.value
6: buf = &buf[i]
7: = s

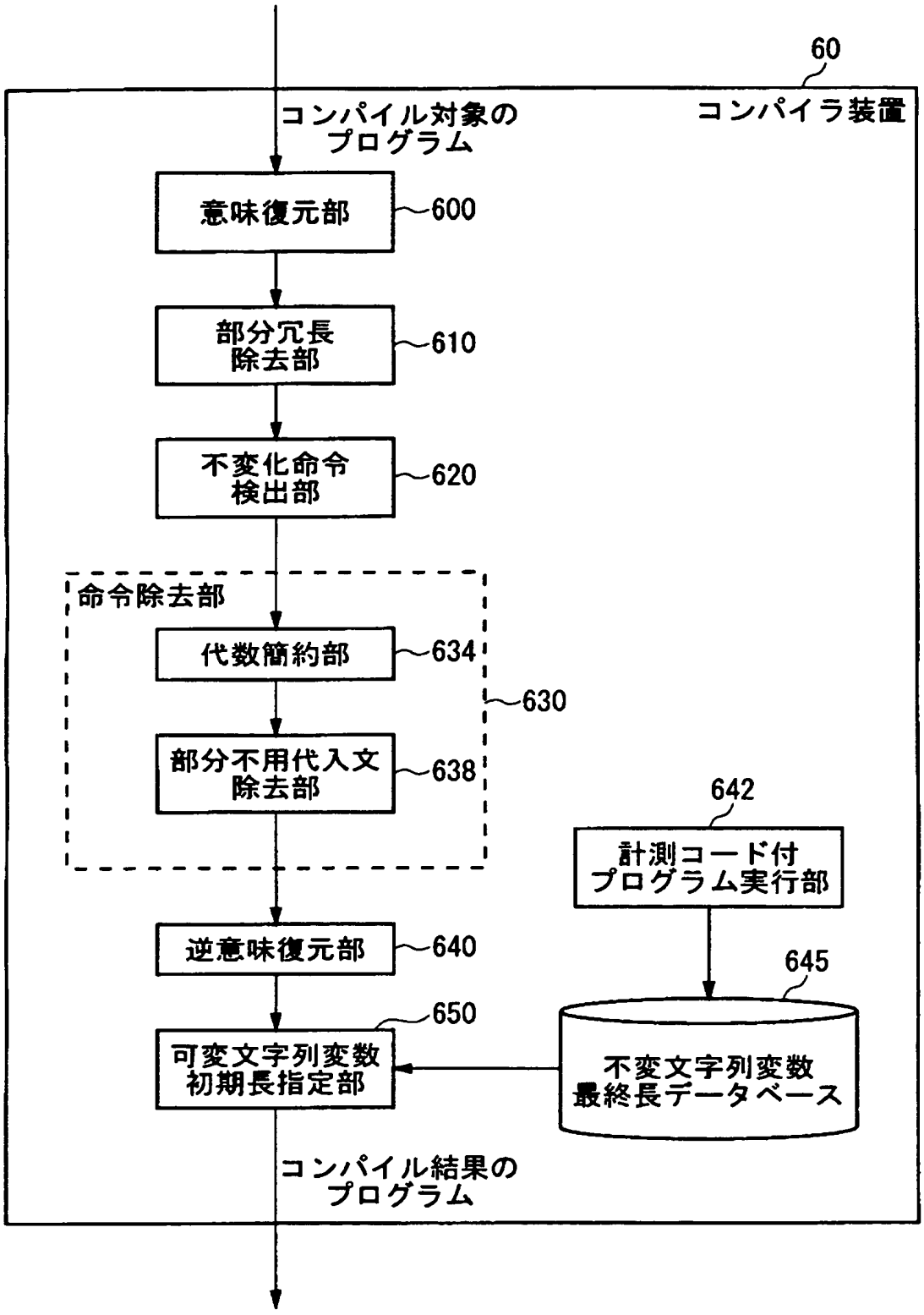
【図４】


1: bufの初期化
2: store c to buf[ ]
3: buf = buf → next
4: store c to buf[ ]
5: buf = buf → next
6: store c to buf[ ]
7: buf = buf → next
8: bufのリストをs.valueに変換
9: = s

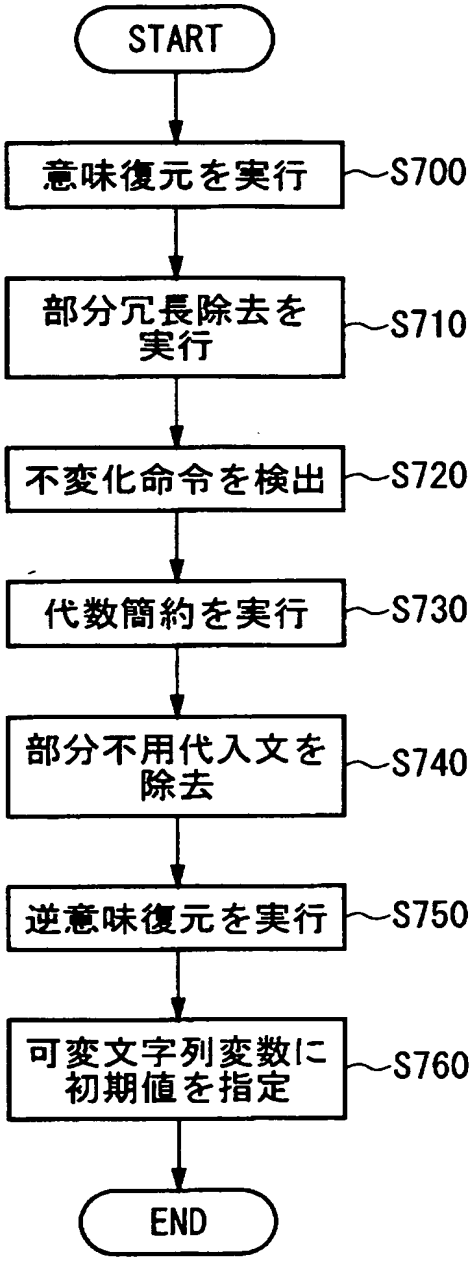
【図５】


(a)
1: sb = s.toStringBuffer()
2: sb.append(c)
3: sb.append(c)
4: sb.append(c)
5: s = sb.toString()
6: = s
(b)
1: s =
2: s = new String(s, c)
3: s = new String(s, c)
4: s = new String(s, c)
5: = s

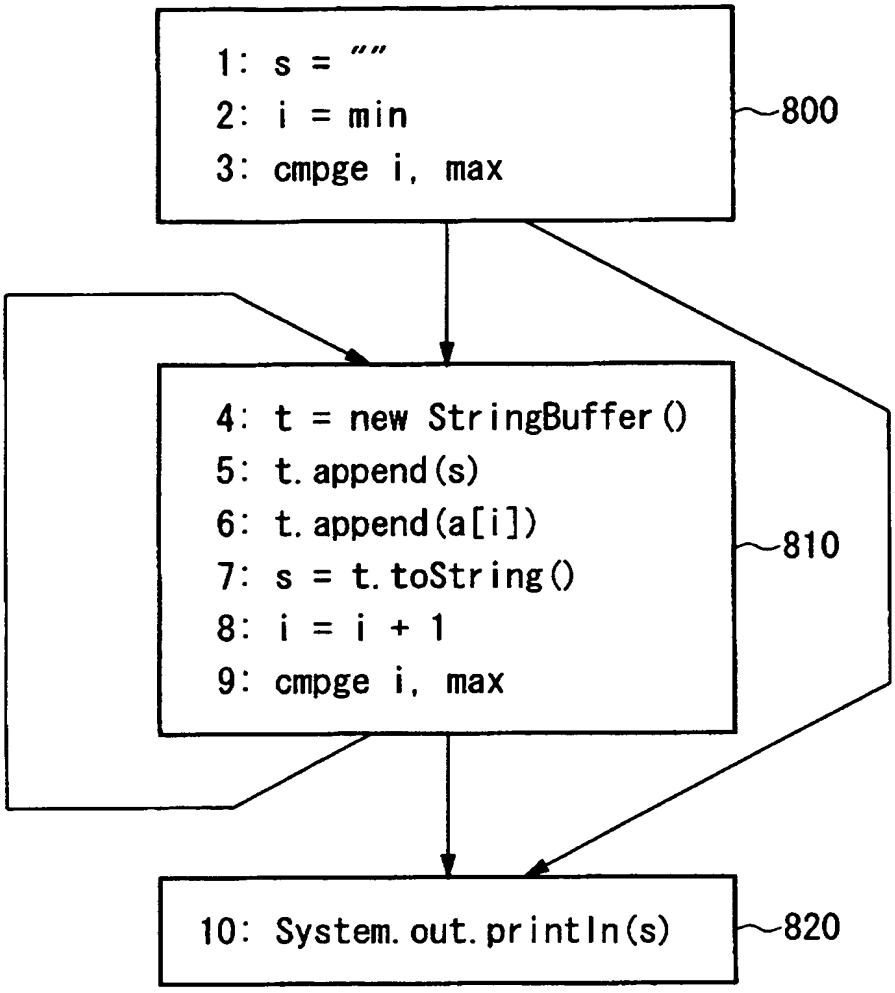
【図６】


60
コンパイラ装置
コンパイル対象の
プログラム
意味復元部
600
部分冗長
除去部
610
不変化命令
検出部
620
命令除去部
代数簡約部
634
630
部分不用代入文
除去部
638
642
計測コード付
プログラム実行部
645
逆意味復元部
640
650
可変文字列変数
初期長指定部
不変文字列変数
最終長データベース
コンパイル結果の
プログラム

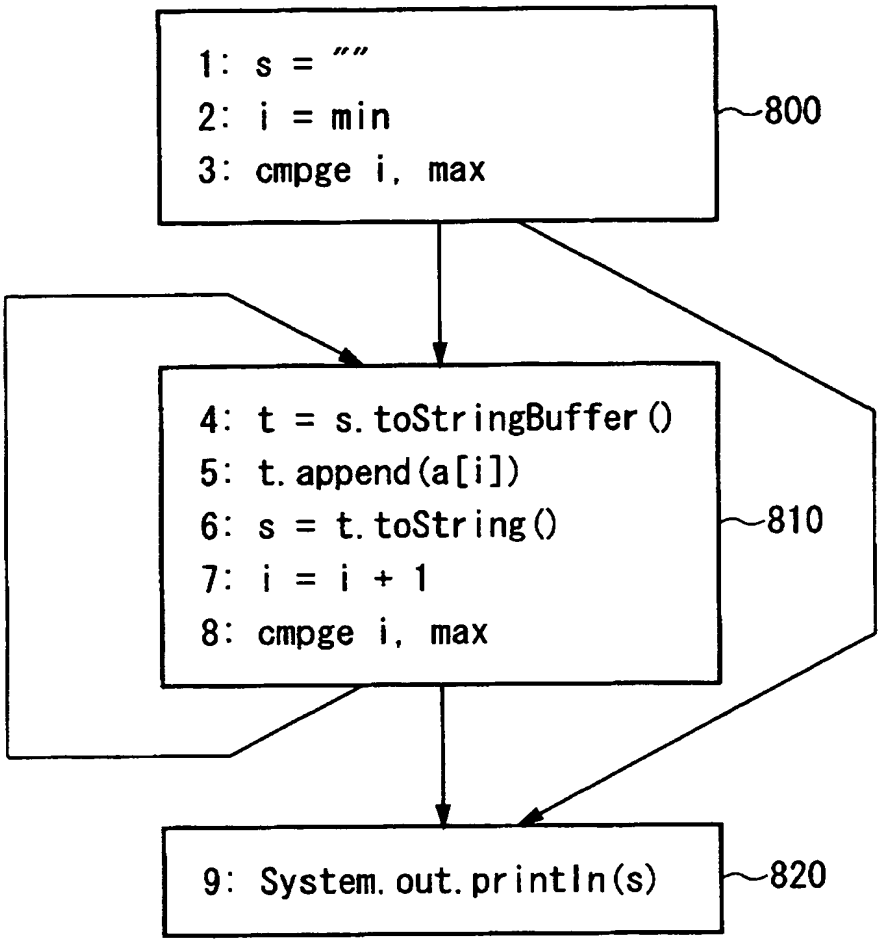
【図７】


START
意味復元を実行
S700
部分冗長除去を実行
S710
不変化命令を検出
S720
代数簡約を実行
S730
部分不用代入文を除去
S740
逆意味復元を実行
S750
可変文字列変数に初期値を指定
S760
END

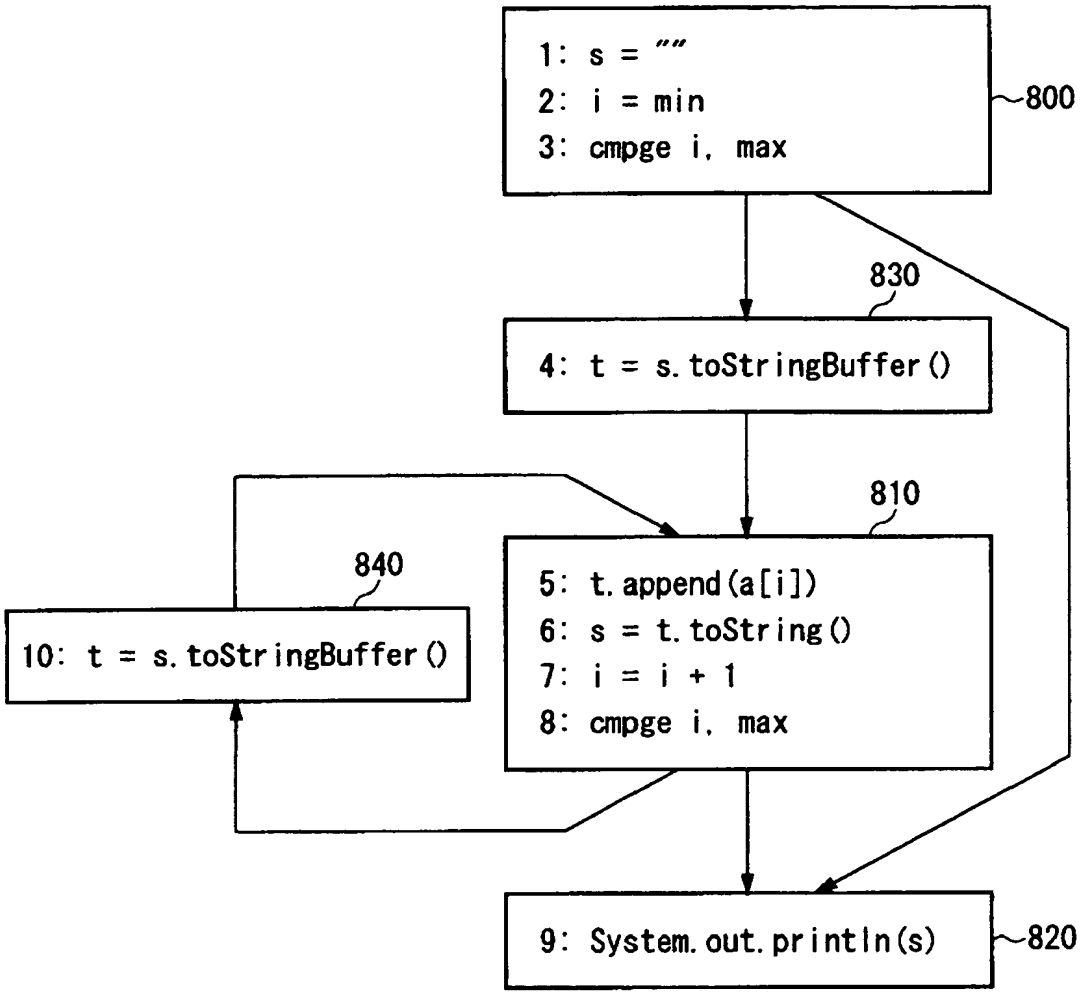
【図８】


1: s = ""
2: i = min
3: cmpge i, max
800
4: t = new StringBuffer()
5: t.append(s)
6: t.append(a[i])
7: s = t.toString()
8: i = i + 1
9: cmpge i, max
810
10: System.out.println(s)
820

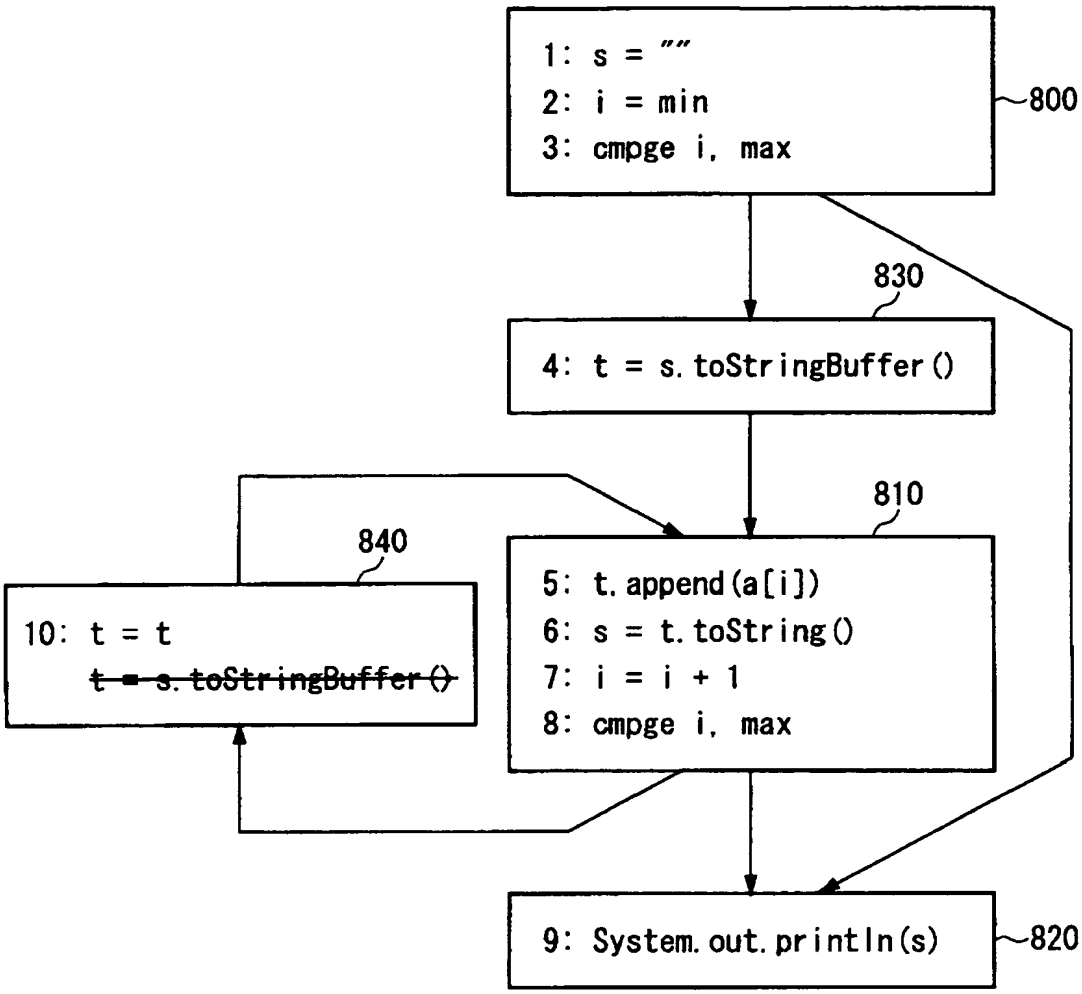
【図９】


1: s = ""
2: i = min
3: cmpge i, max
800
4: t = s.toStringBuffer()
5: t.append(a[i])
6: s = t.toString()
7: i = i + 1
8: cmpge i, max
810
9: System.out.println(s)
820

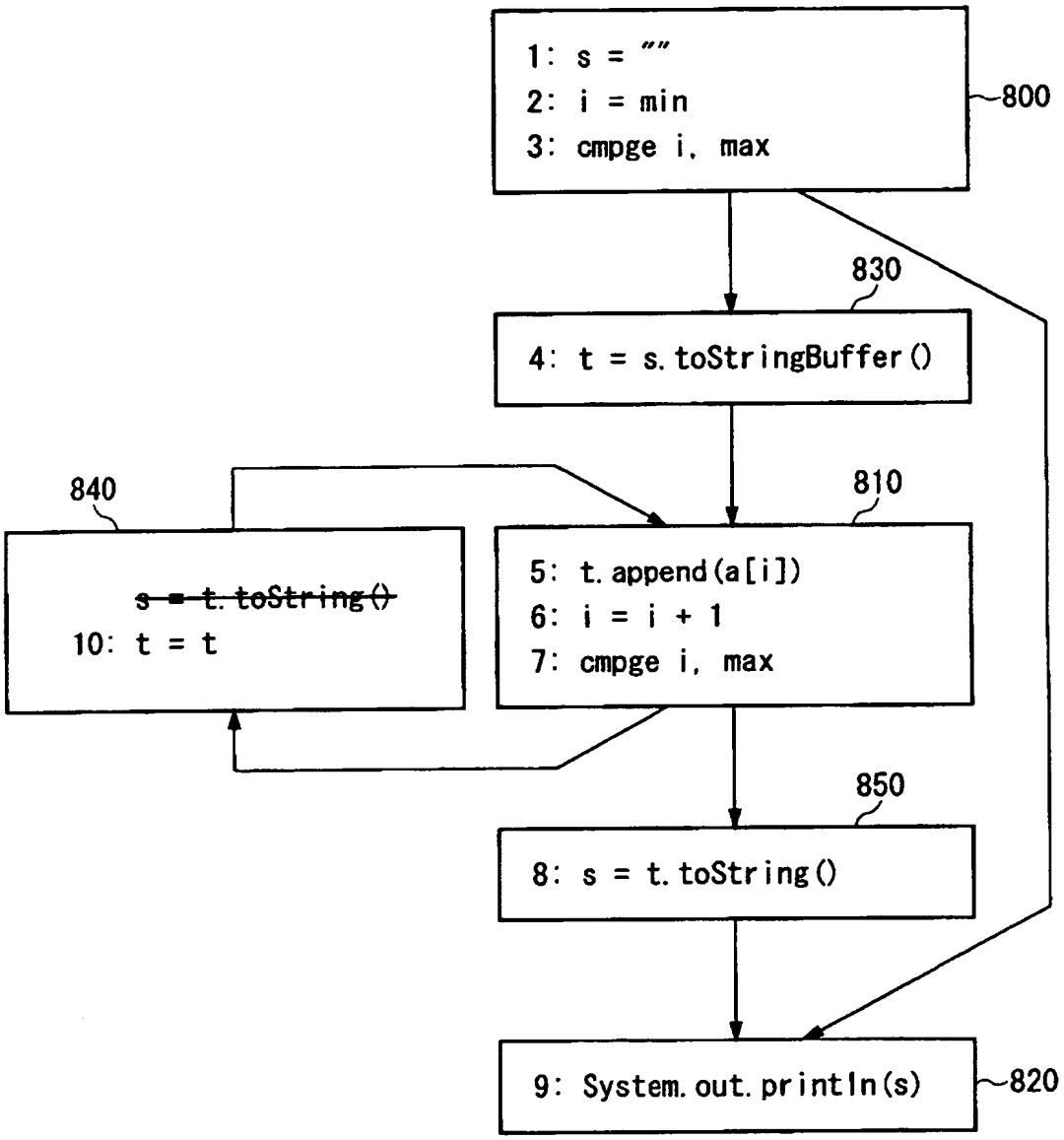
【図１０】


1: s = ""
2: i = min
3: cmpge i, max
800
830
4: t = s.toStringBuffer()
810
840
5: t.append(a[i])
6: s = t.toString()
7: i = i + 1
8: cmpge i, max
10: t = s.toStringBuffer()
9: System.out.println(s)
820

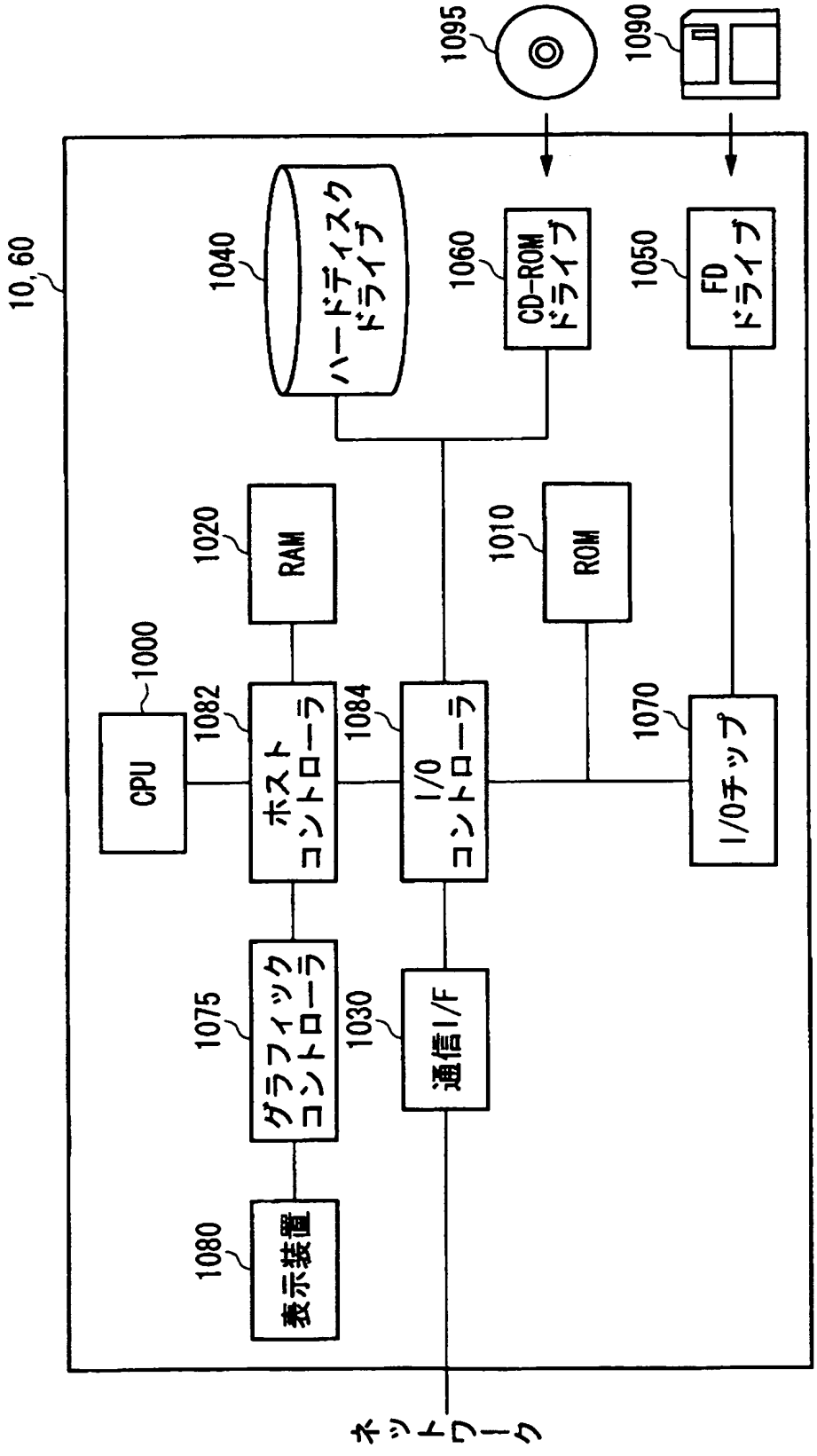
【図１１】


1: s = ""
2: i = min
3: cmpge i, max
800
830
4: t = s.toStringBuffer()
810
840
5: t.append(a[i])
6: s = t.toString()
7: i = i + 1
8: cmpge i, max
10: t = t
t = s.toStringBuffer()
9: System.out.println(s)
820

【図１２】


1: s = ""
2: i = min
3: cmpge i, max
800
830
4: t = s.toStringBuffer()
840
810
s = t.toString()
10: t = t
5: t.append(a[i])
6: i = i + 1
7: cmpge i, max
850
8: s = t.toString()
9: System.out.println(s)
820

【図１３】


10,60
1095
1090
1040
ハードディスクドライブ
1060
CD-ROMドライブ
1050
FDドライブ
1020
RAM
1010
ROM
1000
CPU
1082
ホストコントローラ
1084
I/Oコントローラ
1070
I/Oチップ
1075
グラフィックコントローラ
1030
通信I/F
1080
表示装置
ネットワーク

【書類名】　　要約書

【要約】

【課題】　文字列を操作するプログラムを最適化する。

【解決手段】　文字列を操作するプログラムを最適化するコンパイラ装置は、プログラムにおいて、文字列を格納する文字列変数に文字列を追加する追加命令を検出する追加命令検出部と、追加命令検出部により検出された、同一の文字列変数に文字列を追加する複数の追加命令のそれぞれに代えて、当該追加命令により追加される追加文字列のデータをバッファに格納する格納コードを生成する格納コード生成部と、プログラムにおける、文字列変数を参照する命令より先に実行される個所に、複数の追加文字列のそれぞれを文字列変数に追加する追加コードを生成する追加コード生成部とを備える。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００３－０４９４１４
受付番号	５０３００３１０５３０
書類名	特許願
担当官	伊藤　雅美　　２１３２
作成日	平成１５年　４月　８日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】 390009531
- 【住所又は居所】 アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
- 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】
- 【識別番号】 100086243
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 坂口　博

【代理人】
- 【識別番号】 100091568
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 市位　嘉宏

【代理人】
- 【識別番号】 100108501
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社　知的所有権
- 【氏名又は名称】 上野　剛史

【復代理人】 申請人
- 【識別番号】 100104156
- 【住所又は居所】 東京都新宿区新宿１丁目２４番１２号　東信ビル６階　龍華国際特許事務所
- 【氏名又は名称】 龍華　明裕

次頁無

出　願　人　履　歴　情　報

識別番号　〔３９０００９５３１〕

１．変更年月日　２００２年　６月　３日
［変更理由］　住所変更
住　所　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
氏　名　インターナショナル・ビジネス・マシーンズ・コーポレーション